Emulation – a viable preservation strategy

Authors	: Jeffrey van der Hoeven, Hilde van Wijngaarden (KB)
	: Remco Verdegem, Jacqueline Slats (Nationaal Archief)
Organisations	: The National Library of the Netherlands/Koninklijke Bibliotheek
	: Nationaal Archief of the Netherlands

Date of creation	: 18 January 2005
Date of last update	: 20 June 2005
Version	: 1.4

I. Table of contents

1. Introduction

Digital preservation does not end with the careful storage of digital objects, the union of all objects that are born-digital or digitised like digital documents, electronic records and software programs. Keeping them accessible requires a continuous R&D effort working out strategies for permanent access. The usability of digital objects is threatened by the rapid innovations in computer technology. New systems and software supersede each other faster every year, making existing technology obsolete. This is becoming a problem in everyday life, but is experienced more specifically in the cultural heritage sector, whose main focus is the long term.

Permanent access strategies can roughly be divided into two groups: migration and emulation. Migration is aimed at the digital object itself, and aims at changing the object in such a way that software and hardware developments will not affect its availability. By changing or updating the format of an object, it is possible to render these objects in a current hard- and software environment. Emulation does not focus on the digital object, but on the original environment in which the object is rendered. It aims at recreating an environment in which the digital object can be rendered in its authentic form.

The choice for a strategy depends on several different arguments. First of all, the strategy should be consistent with the objectives of the organization. The Koninklijke Bibliotheek (KB) states that every publication it preserves should remain accessible and interpretable in the way it was originally meant by the publisher. In the case of electronic records archived at the Nationaal Archief of the Netherlands, the requirements are determined by the business process in which the record played a role and by the requisite legal context (see Archival Law 1995 and the Regulation on the Arrangement and Accessibility of Records [Archiefwet]).
A second aspect of the decision for a certain strategy depends on the requirements of the future user. Defining all requirements at the time of preservation is impossible, but we can try to consider what future user scenarios will be. When a future user wants to browse through an old website, the representation plays an important role. This representation makes the digital object and is defined by five aspects: content, structure, context, appearance and behaviour, as shown in figure 1.1. When an object should be recreated to be experienced as it was in the past, the object should be reconstructed as accurately as possible. For this, we need the original environment, which can only be recreated using either original hard- and software or an emulator.

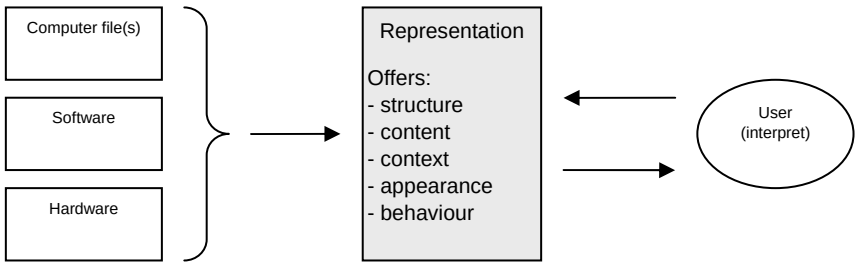

Figure 1.1: Representation model

Finally, the strategy is related to the type of digital object. For relatively simple digital objects like text documents or images, migration (in the form of format conversion or version upgrades) could work, although errors may appear. Migration is often denoted as a strategy for those objects with a short lifetime, with a maximum of ten years but usually only a few. The risk that subsequent migration cycles further decrease the authenticity of the object makes this strategy less suitable for the long term. For more complex digital objects, like advanced text documents and spreadsheets including formulas, migration will lead to notable changes of the original object. Moreover, migration does not preserve the hard and software environment in which the objects were originally created. This environment is an integral component of the representation process of the object as we experience it. Functionality offered by the software environment makes it for example possible to scroll and search through text, click on a cross-reference or run a query on a database. As digital objects are increasingly dependent on functionality, ignoring the environment disables the possibility of full reconstruction of the objects. Especially digital objects that depend strongly on the original software environment can not be authentically rendered by another approach than emulation.

The choice for emulation as a preservation strategy is not undisputed, even though its possibilities are recognized. Whatever the different views may be, one fact remains: emulation has never been actually developed and tested within an operational digital archiving environment. The National Library of the Netherlands, Koninklijke Bibliotheek (KB) and the Nationaal Archief of the Netherlands, apart from being convinced that emulation is possibly the only option for future access to digital objects with preservation of their functionality, also strongly believe that we have to develop and test this strategy first, before we can evaluate it.

This is the reason the KB and the Nationaal Archief will start a project to develop an emulator for digital preservation. This document describes the current situation in both institutions, a short overview of developments in emulation, issues and scope of the project and results of an investigation into possible scenarios. It ends with a description of practical steps to be taken in the project to develop emulation-based preservation.

2. Current situation

2.1. The National Library of the Netherlands

As National Library of the Netherlands, the Koninklijke Bibliotheek (KB) is responsible for the Dutch deposit library. The aim of the deposit library is to collect, catalogue and preserve all publications appearing in the Netherlands. This task includes publications on paper as well as in electronic form. In order to meet the challenge of electronic storage, the KB, together with IBM, has developed a large-scale storage system that meets the specific requirements of long-term preservation: the e-Depot. In addition to long-term storage, permanent access strategies have to be worked out. The KB has established a research and development program to develop innovative tools and procedures to ensure future accessibility of stored digital publications.

The majority of electronic publications in the e-Depot are deposited and stored in Portable Document Format (PDF). For a long time, PDF documents were fixed-format text documents. However, newer versions of PDF can contain all kinds of embedded formats, in-text cross references, forms and even pieces of scripting code that make PDF more and more a dynamic format [Ockerbloom]. In addition to PDF, other formats are accepted for storage in the e-Depot as well, e.g. Office-documents and image formats.

Since the e-Depot became operational in 2003, another type of digital objects has been stored as well: interactive multi-media applications, which are supplied on CD- or DVD-ROM by publishers. These applications are installed on a standardized computer platform, which is called a Reference Work Station (RWS, for the specification, see appendix A). After installation, a snapshot is taken from the hard disk's content by creating a disk-image of the installed application including the operating system. This image is stored in the e-Depot and can be reinstalled on the RWS at any time.

To retrieve the stored digital objects in the future, the e-Depot has to be extended with new functionality. Using the RWS for the rendering of interactive multi-media applications is just one strategy which will buy us some time to work on another solution. PDF documents and other file formats that are stored are still accessible, but we have to work on rendering strategies to enable future access. The KB aims at keeping publications accessible in the format as they were delivered to the KB by publishers. Migration may alter the appearance of a publication which also may change its meaning. Migration does not offer a solution for interactive and complex digital objects either. Therefore, the KB will invest in the development of an emulator for preservation purposes.

2.2. The Nationaal Archief of the Netherlands

The Nationaal Archief of the Netherlands has officially existed in its current form since 4 June 2002. But the national archives service in the Netherlands is now 200 years old. The Nationaal Archief is the leading centre for the study of Dutch history and culture at the national level. As the "national memory", the Nationaal Archief manages not only historic government records but also the archives of private individuals who play or have played an important part in the life of the nation. It is the task of the Nationaal Archief to gather these archives, to maintain them in good, ordered and accessible condition, and to present them to a broad public.

Objectives
The formal status of the Nationaal Archief is enshrined in the 1995 Archives Act. This designates the Nationaal Archief as the central repository for archives transferred from national government bodies. That is, those institutions which have or had functions encompassing the Dutch nation as a whole. In particular, this definition refers to the High Councils of State, the Queen's Cabinet and the ministries.
The Nationaal Archief also has an important public function. This is defined in its mission statement*: "The Nationaal Archief supplies historical information to a varied public, based upon the content of its collection: the archives related to national government."* This wording reflects the fact that the needs of users are considered alongside the Archive's statutory function. Information is made available and accessible, and is supplied to the public.

Policy
The Nationaal Archief has formulated a number of ambitions for the next few years. These are set out in the policy statement *'Geschiedenis binnen handbereik'*. At the heart of our ambitions are efforts to make our collection more visible and accessible, and to provide a broad and varied public with a greater insight into the history of the Netherlands. The most important social issues are promoting a transparent and controllable government and maintaining the cultural heritage. As the Nationaal Archief gives access to ‘authentic’ sources, the Nationaal Archief is able to address these social issues. The dynamics of the digital developments requires a new vision on the future management and use of digital collections. Sustained accessibility, continuity and linkage of digital sources are important conditions for ‘future proof’ public services. Digital cultural heritage and electronic records lend themselves admirably to the purpose of applications which serve these public services. Condition is that the digital records are accessible in a sustainable way.

The Nationaal Archief of the Netherlands has the legal task to secure the sustained accessibility of the records of the government. This applies to paper records as well as to electronic records. For paper records good facilities are in place. In order to meet this challenge for electronic records, which the Dutch government is producing increasingly, the Nationaal Archief is implementing a digital depot for the storage and sustained accessibility of digital records.

2.3. What is emulation?

From 1999 until 2003 the CAMiLEON project (Creative Archiving at Michigan and Leeds: Emulating the Old on the New) conducted research into the possibilities of emulation as digital preservation strategy [CAMiLEON (1)]. They defined emulation as: “the re-creation on current hardware of the technical environment required to view and use digital objects from earlier times.” [CAMiLEON (2)] Emulation is the process of bringing digital objects back to life in their original environment on top of a different computer environment. This process is carried out by an emulator, which by definition of the Digital Preservation Testbed “is a program that runs on one computer and thereby virtually recreates a different computer.” [Testbed 2003] In this definition the word virtual denotes that the emulator functions like the original computer, but physically is not. The original computer is called the target platform; the computer that executes the emulator is called the host platform.

2.3.1. Emulation levels

Emulation can be done at three different levels: application software level, system software (operating system) level and hardware level. Emulating both application and system software requires knowledge of their design and implementation. These products are complex and very often proprietary, which makes it difficult to emulate. Another issue with application level emulation is that each application requires a specific emulator.

Emulation can also be done by mimicking the hardware architecture through software, which is called *software-emulation-of-hardware* or *full emulation*. In this way, computer hardware like a processor is emulated by a software surrogate. In fact, the emulator is placed on top of the host operating system (OS), instead of running directly on hardware. This allows us to take advantage of the services and flexibility offered by the host OS, like the use of its Application Programming Interface (API). An emulator is therefore a software program that runs on a host platform (hardware and OS) and recreates the hardware of the target platform under emulation, as depicted in figure 2.1.

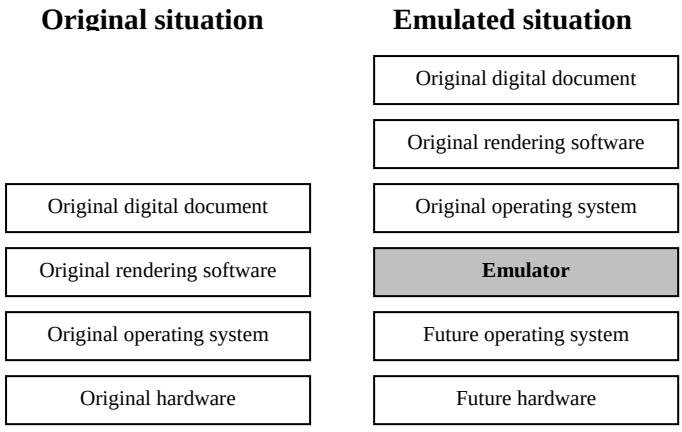

Figure 2.1: Software-emulation-of-hardware

Although full emulation can also be quite complicated, it has a more straight forward behaviour than emulating higher levels. For example, higher level features such as graphical user interfaces and running multiple applications side by side (known as multi threading) are difficult to emulate accurately. Emulation of a hardware platform does not incorporate these aspects, but requires the reproduction of the functional behaviour of the original platform in such a way that the original

software is not able to distinguish the difference between emulation and reality. Because hardware specifications are well defined and most often available, this behaviour is easier to reproduce than that of an OS or software application. Moreover, this approach retains the original OS, applications, drivers and configuration, which secures better authenticity of the original environment.

It is also possible to emulate hardware with other hardware [Testbed 2003]. This is frequently done to create simpler or lower-power versions of existing computer processors, though it can just as well be done to create faster or more powerful versions of popular older machines. From a preservation perspective, this is quite similar to software-emulation-of-hardware, since it can be used to recreate a given obsolete hardware platform on which original rendering software for obsolete formats can be run in the future. Its disadvantage is merely that it is often (though not necessarily) harder and more expensive to build a hardware emulator of this sort than to write an emulator program. Nevertheless, this approach might in the future become competitive with the software-emulation-of-hardware approach discussed here.

A slightly different approach of emulation is the use of a compatibility layer [Wikipedia (1)]. A compatibility layer translates system calls of the host operating system into system calls for the target system and vice versa. The calls are defined in a library, called Application Programming Interface (API) of the host operating system. The compatibility layer makes use of the existing API of the host operating system and offers better performance as with normal emulation. However, a compatibility layer is very dependent on the underlying operating system and appropriate API, which makes it very vulnerable over time. Furthermore it is not based on the principles of reconstructing the original environments and objects, but merely gives an alternative representation in a current environment. Examples of compatibility layers are: the Linux compatibility layer on FreeBSD, WINE, and MS Windows XP emulating MS-DOS and MS Windows 98.

In the rest of this document emulation stands for emulation of hardware by means of software. An emulator is therefore a software program that runs on a host platform (hardware and OS) and recreates the hardware target platform under emulation.

2.3.2. Practices

Emulation is not new. It has been applied for years already and for many purposes. Numerous emulators are written and can be found on the Internet [Wikipedia (2)].

The available emulators can roughly be categorized by four purposes:

1. Game computer platforms
2. Historic computing
3. Business efficiency
4. Cross platform emulation

Game computer platforms
Since the introduction of the game console decades ago, many generations have followed. Because the rapid speed of technology and the lack of compatibility between subsequent platforms, many of them have become obsolete over time. The nostalgic aspect of games experienced by users has led to actions to keep older game console platforms alive by means of emulation.
Examples of platforms : Atari, Nintendo, Sega and other arcade machines.
Examples of emulators : MAME

Historic computing
The same nostalgic aspect can be found by running old computer platforms. Like emulators for game consoles, several emulators exist for historic computer platforms.
Examples of platforms : Commodore, MSX, PDP.
Examples of emulators : SIMH

Business efficiency
Emulation can also lead to more efficiency in business environments. Emulators have been created that make it possible to run multiple operating systems on the same hardware. This way, the infrastructure is used more efficiently and it offers the possibility to test and debug applications in a safe environment before releasing them to the public. Several corporate emulators are available.
Examples of platforms : x86, PPC and SPARC
Examples of OS : Windows, Linux, MacOS
Examples of emulators : VMware, MS Virtual PC, SVISTA

Cross platform emulation
Emulators are also built to host different platforms on other platforms, i.e. cross platform emulation. Especially in the open source community where Linux compatibility towards other operating systems is desirable, numerous initiatives can be found.
Examples of platforms : x86, PPC and SPARC
Examples of OS : Windows, Linux, MacOS, BSD, BEOS, Reactor, DOS
Examples of emulators : Bochs, QEMU, PearPC

3. The emulation project

This chapter has been written in cooperation with Jeff Rothenberg, RAND Corporation.

The emulation project is a proposed joint project by the National Library of the Netherlands (KB) and the Nationaal Archief of the Netherlands to develop a preservation strategy for digital objects using emulation. In the following sections this project will be outlined. This includes the goal, scope and choices to be made.

3.1. Criteria for the project

The project will be a first phase towards the implementation of an operational emulation based solution for the accessibility of digital objects in their original environment. This first phase serves the following goal:

> "Produce a working prototype emulation preservation capability which can serve as the basis for further, more robust versions. This prototype should provide an actual preservation capability within its chosen scope. Although the emulator should ultimately provide an operational solution for permanent access within the digital repository infrastructure of the KB and the Nationaal Archief, the first phase should not be constrained to developing a production tool, since this would restrict its ability to explore alternative implementation choices."

With this working prototype it will become possible to:

- Evaluate the use of emulation to preserve current digital artifacts in a manner that will allow them to be rendered in their original forms on future platforms.

- Evaluate how a single emulation approach can preserve several digital object types (formats), each of which is of significant value to the KB and/or the Nationaal Archief.

3.2. Dimensions of Scope

The scope of the proposed project can be defined by the following 5 dimensions:

1. Format/Object types
2. Target platform
3. Significant properties
4. Host platform
5. Enabling future reuse

3.2.1. Format/Object types

It is important to target more than one object type in order to demonstrate the potential universality of the emulation approach and to avoid specialized solutions based on the limitations of any single

format. A small number of different formats seems most appropriate for a first phase. Candidate formats might be those that:

a) Are used by large numbers of items in the KB's or National Archives' collections
b) Are of vital importance to the KB or National Archives (regardless of their number of items)
c) Appear to pose particular preservation or emulation challenges

A possible subset of object types, each of which satisfies one or more of these criteria are:

* Interactive multi-media applications
* Database systems
* PDF documents

Interactive multi-media applications

CD- / DVD-ROM publications contain interactive multi-media applications that exhibit a wide range of behaviour, such as sound, animation, video clips, database access, etc. They therefore serve as a surrogate for the wider range of interactive applications, while providing much of the behaviour of web applications without involving the complexity of the network. The KB currently has about 700 interactive multi-media applications stored from CD-ROM as "installed images" which should be relatively straightforward to run under emulation.

PDF documents

Currently the e-Depot of the KB contains millions of electronic publications formatted in Portable Document Format (PDF). The Nationaal Archief of the Netherlands receives nearly all *textual* documents in PDF-format. To maintain access to these documents different strategies can be applied. Although the UVC-approach (as will be further outlined later on in this document) guarantees the reconstruction of the original representation of a document, it loses any functionality like full-text search, internal and external linking, interactive forms and added notes and bookmarks.
PDF represents the mainstream of both the KBs and Nationaal Archief's current digital holdings. Although it should be a relatively simple case for emulation, its importance makes it an attractive candidate.

Database systems

A database system consists of three components

- the database itself (the actual content);
- a DataBase Management System, DBMS (for example, Oracle 9i);
- the database application. This incorporates both the graphical user interface and the functionality the user needs to search through and process the content of the database, as well as programs that function automatically to support the system in processing inputs and outputs.

They may be normally configured as distributed client-server or multi-tier applications, though we assume that these could be reconfigured to run on a single standalone workstation.

Currently, the Nationaal Archief of the Netherlands uses migration and XML to preserve databases. The migration of databases from an older version to a newer version of the same database system (e.g. Access 97 to Access2000) can be used to represent context, content, appearance, structure and behaviour for the short term (less than 10 years). The conversion to XML is suitable to represent the context, content and structure of the database itself. Additionally, in order to preserve the appearance of the application it is necessary to store the technical and functional documentation of the database system, including screen shots. We were not able to preserve behaviour of database systems for the longer term using migration or XML. Nor is the UVC data preservation approach able to achieve this.

Emulation could be the only approach in this respect, but has not been implemented with an archival focus.

Once a set of objects is chosen for the project, a range of example items in those objects should be sampled to determine their significant properties, e.g., how much display resolution, colour accuracy, sound fidelity, etc. they require.

3.2.2. Target platform

By "target" we mean the platform on which the original artifacts run. The choice of objects should be used to determine the required characteristics of a minimal target platform. For example, if distributed DB system can be configured to run on a standalone workstation, then such a workstation might be a suitable target platform. We recommend avoiding a target platform that involves distributed architecture or networking in the first phase of this project.

3.2.3. Significant properties

These are the interactions that must be considered, such as static or dynamic visual display, sound, etc., along with the required level of fidelity with which each significant property (also called modality) needs to be reproduced. A minimal set of such properties with minimal levels of fidelity should be determined based on the chosen set of object types. Examples of likely target property fidelities required for the first phase of the project are:

* Standard SVGA or better screen resolution
* CIELAB colour
* 44KHz stereo sound
* Processing speed near that of the target platform

Processing speed is included as a significant property because it can greatly affect interaction.
The goal for a first phase emulator should be to at least report the percentage of target platform performance that it can achieve, ideally achieving at least as much as the target platform. The default should be to limit the speed of the emulator to be no more than that of the target platform, though faster speed may also be an option.

3.2.4. Host platform

The main issue here is whether to write the emulator to run directly on a real platform or on a Virtual Machine (VM). The latter is highly desirable for portability purposes, but it may reduce performance unacceptably, and it may limit the emulator's ability to achieve required levels of fidelity for various significant properties. For example, an emulator written in Java (to be hosted on the Java Virtual Machine, or JVM) may not be able to control the native colour or sound capabilities of the real platform on which it runs. Besides the JVM, more general VMs are also of interest, like the Universal Virtual Computer (UVC) of Raymond Lorie or the Emulation Virtual Machine (EVM) as suggested by Jeff Rothenberg.

Although we consider it desirable to host emulators on a VM, it may be better to defer this goal to later phases, first getting an emulator running on a native platform. This should not mean that a lot of

work will have to be repeated to take the next step towards running the emulator on a VM. Ideally, the emulator would be written in a language that could be compiled either directly to machine code or to run on a VM. One possible candidate for this could be Java, for which compilers exist that can convert Java source code into JVM bytecode or directly into platform dependent native code. Another candidate language could be Ada, for which compilers exist that can produce either native machine code or JVM bytecode.

3.2.5. Enabling future reuse

This dimension concerns the provision in the emulator (or an emulation environment, such as a VM) of features that can extract content from an original artefact that is running under emulation, enabling that content to be reused in some future "vernacular" format. This can be thought of as "vernacular extraction" or as "migration on demand". It is not yet clear exactly how an emulator or emulation environment can best support such vernacular extraction, but it could be important that the emulation approach be able to do this. Therefore, although full inclusion of this feature will be postponed until later phases of the project, it may be useful to address its possible requirements already in the first phase--at least to the extent necessary to ensure that nothing in the chosen emulation approach precludes providing this capability in the future. Part of this effort might include an attempt to predict the kinds of reuse capabilities that future users might want, although we realize that this may be difficult to ascertain.

3.3. Set of choices

Decisions must be made along each of the above scope dimensions. However, a tentative set of choices might be:

1. Object types
   * Interactive multi-media applications
   * Database systems
   * PDF documents

2. Target platform
   * Standalone Pentium-based workstation (the predefined ‘Reference Work Station’ at the KB)

3. Significant properties
   * Standard SVGA or better screen resolution
   * CIELAB colour
   * 44KHz stereo sound
   * Processing speed near that of the target platform

4. Host platform
   * VM or native

5. Enabling future reuse
   * Avoid precluding such capabilities

4. Emulation research

During October 2004 until March 2005 an investigation has been done in the field of emulation as a prologue for the emulation project carried out by the KB and Nationaal Archief. In this research, a number of issues were addressed as: what kinds of emulators are already available? What is the purpose of it? Which approaches of emulation in the field of digital preservation are discussed? Which scenario or scenarios can be applied by the KB and Nationaal Archief or can we define a new approach of emulation?
In the following sections these issues are further discussed.

4.1. Evaluating existing emulators

Various emulators available on the Internet were tested and evaluated. It was remarkable how many different emulators exist for many different platforms. In the search for emulators available on the Internet, certain specification requirements of the RWS hardware platform have to be met. In short, it has to be compatible with an x86 platform with a certain amount of internal memory and hard disk space, a compatible display adapter, sound card and networking capabilities. Furthermore it should be able to attach and boot from a disk image. Several candidate emulators were found:

- MS Virtual PC
- VMware
- QEMU
- Bochs

MS Virtual PC
The MS Virtual PC emulator is a fast and all-round emulator developed by Microsoft [Virtual PC]. The emulator was initially developed by Connectix that offered an x86 emulator especially for the Macintosh. In this way, Macintosh users were able to run a Microsoft operating system with full functionality under a Macintosh environment. Today, the Virtual PC solution is property of Microsoft and allows emulation under MacOS or MS Windows itself so that multiple Microsoft operating systems can run side by side or even on top of each other (layered emulation). It offers a great performance, but uses specific disk images that are not interchangeable. Another disadvantage is that the implementation is bound to an MS Windows environment, which lacks support for other host and target platforms (except for MacOS).

VMware
VMware is a virtual machine (VM) suite for Intel x86-compatible architectures which allows the creation and running of multiple virtual x86 computers simultaneously, which can run a number of guest operating systems (OS), including (but not limited to) Windows, Linux, and BSD variants [Wikipedia (3), VMware]. Conventional emulators emulate the entire PC hardware including the CPU. VMware takes a different approach, adding a thin layer of code to virtualize the real PC hardware and CPU, so that multiple operating systems can run at the same time without clashing with each other. Besides VMware, Microsoft also has developed such virtual machine software, called Microsoft Virtual Server 2005. This serves the same principle as VMware [MS Virtual Server].

The benefit of this approach is that an OS running inside VMware can run at almost the same speed as it would if it was the only OS on the system. For businesses VMware is an attractive solution, because it allows better utilization of the hardware infrastructure. The drawback is that the OS has to be

compatible with your physical hardware. So unlike with an emulator, you cannot use VMware to run Macintosh software on a PC, or vice versa. Moreover, both VMware and Virtual Server 2005 are corporate products that are not developed for digital preservation purposes or cross platform compatibility. This makes it less reliable for a long-term solution.

QEMU
QEMU is a fast and open source cross-platform emulator. It can support x86, PPC, ARM and SPARC computer platforms on many different host platforms [QEMU project]. The QEMU emulator uses dynamic translation (Just-In-Time compilation) of the code, which offers a good performance level. Some parts of the emulator are based on the Bochs emulator implementation. The advantage is that Bochs and QEMU use compatible 'raw' disk images.

Bochs
The Bochs emulator is quite similar with the QEMU emulator, but only enables emulation of an x86 as target platform [Bochs project]. It is also slightly slower than QEMU, but can support more peripheral devices and is better configurable. As said before, it can interchange disk images with QEMU based on 'raw' data image format.

4.2. Approaches for emulation in digital preservation

In general, a couple of different approaches for emulation in digital preservation context are discussed. The central issue in this is how to ensure that an emulator written today can still run in the distant future. In the following table four primary approaches are outlined that can be found in literature:

1.	Stacked emulation	Platform dependent emulation that requires multiple emulators running on top of each other to reconstruct a historic platform.
2.	Migrated emulation	Creating a platform dependent emulator that must be adapted (migrated) to subsequent newer host platforms.
3.	Emulation Virtual Machine (VM)	Emulation using a virtual machine and an emulator specification interpreter, as presented by Jeff Rothenberg.
4.	UVC-based emulation	Using the UVC as universal platform on which a platform independent emulator can be build. I/O communication between host platform and UVC is translated to a more abstract level.

Table 4.1: approaches for emulation in digital preservation

In the following sections each approach will be further discussed.

4.2.1. Stacked emulation

With normal, platform-dependent, emulation a particular emulator emulates a specific hardware platform directly on top of a current platform and OS. This form of emulation is most common for today's emulators. The advantage of applying this kind of emulation is the efficiency that can be gained by specialising the emulator for one particular host platform. This generally brings better performance and functional behaviour of the emulated platform. However, it lacks compatibility between other platforms and is less durable because the emulator is bound to one particular configuration of the platform. Changing the platform by hardware and / or software probably results in a malfunction of the emulator. Examples of platform-dependent emulators can be found in the gaming industry for Atari, Nintendo and Sega emulators, but also more general purpose emulators for platforms as C64, MSX, PPC and x86. These emulators run on a specific hardware platform and OS and recreate a different platform on top of it.

To keep emulators running on subsequent platforms over time, stacked (or layered) emulation can be applied. With stacked emulation the original platform is recreated by a sequence of emulators that form a chain from present to past platform support. Figure 4.1 depicts this process schematically by adding an emulator each time a previous platform became obsolete.

The danger of this approach is that each emulator strongly depends on its underlying platform(s). Functionality that is not available by the platform and intermediate emulators can not be offered to the target platform. Moreover, if one of the in-between emulators is corrupted or missing, the chain relying on this emulator is inevitably lost, resulting in inaccessible applications and documents. A side effect of stacked emulation is the increasing overhead each time an extra emulation layer is added. This generally results in a loss of performance.

Although this approach shows several weaknesses, emulators exist that are capable of performing some kind of stacked emulation. For example, the Virtual PC emulator from Microsoft is able to run older versions of Windows under subsequent layers of emulation, e.g. Windows XP running Virtual PC that runs Microsoft 98, that itself runs Virtual PC and Windows NT on it.

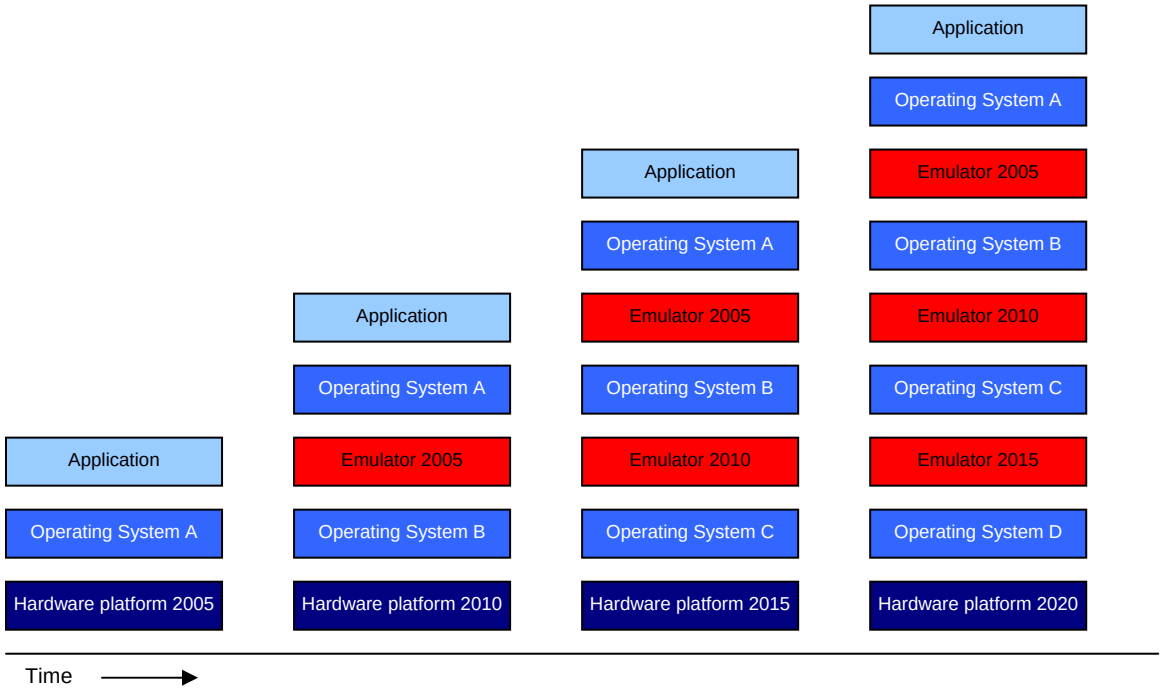

Figure 4.1: Stacked emulation over time

4.2.2. Migrated emulation

When migrated emulation is applied, the emulator is adapted to a different environment. First, an emulator is created for one particular operating system. Then, if the environment of the emulator changes, i.e. the old OS becomes obsolete and a new one is current, the emulator is translated to run in the new environment. This translation process requires a compiler that is able to convert the source code written in language X into a binary executable that will work on the new operating system platform. In this way, the original emulator can be periodically migrated to subsequent environments, as shown in figure 3.2.

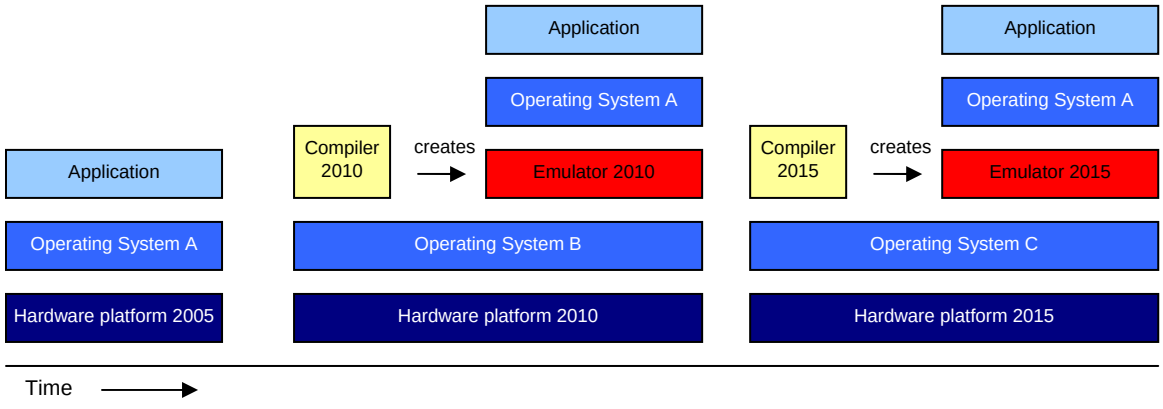

Figure 4.2: migrated emulation over time

However, the source code of the emulator is not compiled today, but in the distant future. This raises some environmental problems because it is not guaranteed that a compiler for today's favourite source language X is still available. And even if it is, the compiler's library may not be compatible with the original one, which can lead to malfunctions.

The CAMiLEON project took an attempt to build an emulator using a subset of the programming language C, which they called C-- [CAMiLEON (3)]. The idea of this approach is that the C-- set only contains basic C functions that can be easily maintained in the future. Although CAMiLEON showed that this approach worked for emulating the old ICL 1900 platform, they also admit that it is inevitable that restrictions of C-- make certain things impossible. Since an emulator is an advanced application that relies heavily on communication with the host and target platform, migrated emulation is a strategy with high risks.

4.2.3. Emulation Virtual Machine

Ideally, an emulator should be independent from time and platform. Jeff Rothenberg defined an approach to reach this independency by introducing an additional layer between the host platform and emulator, called an emulation virtual machine (EVM) [Rothenberg (1)]. Together with an emulator specification and an emulation interpreter, emulators could be created to run on the EVM, as shown in figure 4.3. In theory this approach should work as follows:
An emulation specification describes how the old computer platform worked on which the original software runs. This specification is then interpreted by an emulation specification interpreter that creates an emulator for the old platform. Both interpreter and created emulator run on the EVM, which itself runs on a future platform. The EVM is said to be stable over time and able to run on various host platforms. In this way, an emulator is not bound to any particular platform anymore. Furthermore, it is possible to run multiple emulators on the same EVM.

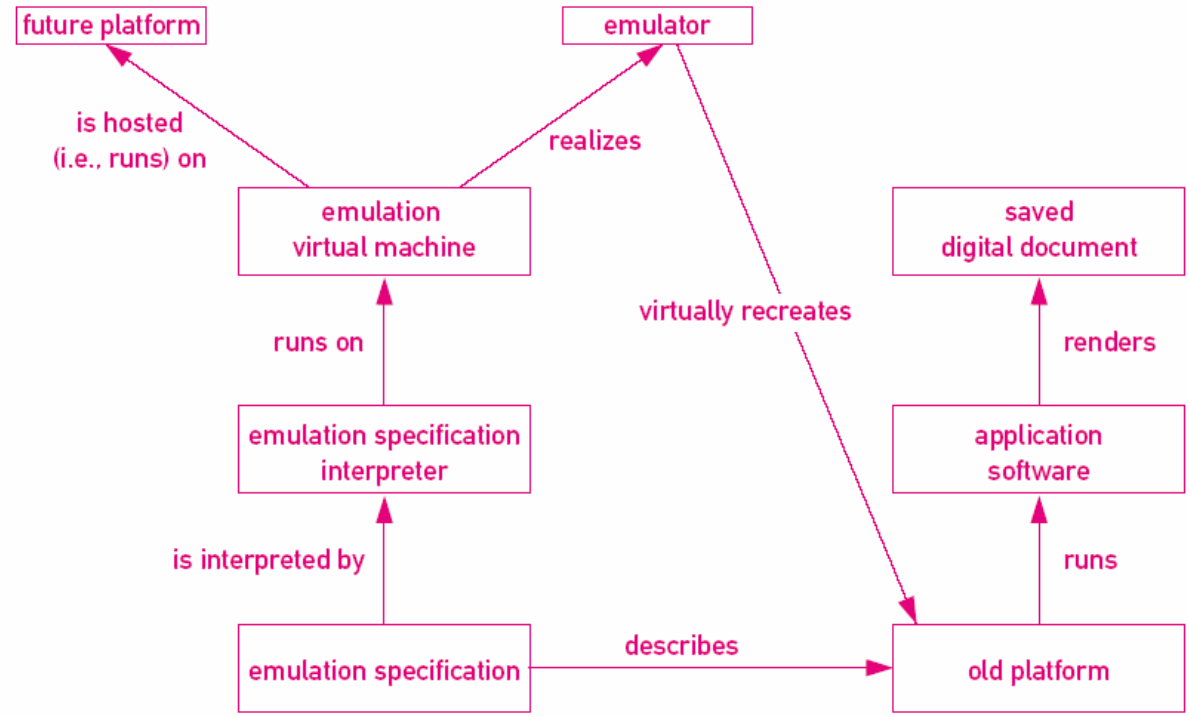

Figure 4.3: schematic representation of the working of the emulation virtual machine approach

Although this approach is said to be platform and time independent, a couple of disadvantages do exist. First, it can be doubted if the old platform can be totally defined in a single specification. Furthermore, a complex interpreter is needed to convert this specification into an emulator. It is questionable if that is possible. Besides that, the EVM must be maintained over time and will also be quite complex.

4.2.4. UVC-based emulation

A variant on the use of a virtual machine is the Universal Virtual Computer (UVC), invented by Raymond Lorie from IBM [Lorie]. Together with IBM, the KB and Nationaal Archief worked on this concept.

The UVC is the basis of the UVC-based preservation method. This preservation method offers permanent access to digital objects in their original representation any time in the future. It is based on a combination of emulation and migration. Migration is done by converting original objects into a platform independent format and emulation by running the process platform independently.
The UVC is actually a software program that recreates a simple general purpose computer and can easily be implemented on any computer platform now and in the future. In this way, programs that are written in UVC language can be executed on the UVC in a platform independent manner. UVC programs (decoders) can now decipher the format structure of the original object. This decoding process delivers an XML-like structure, called Logical Data View (LDV). This LDV is an instantiation of its blueprint: the Logical Data Schema (LDS), which defines the elements that can be expected in the LDV and offers extra information about what the elements mean. The final part is the viewer that reconstructs a view of the original object, based on the LDV elements and LDS description. This process is depicted in figure 4.4.

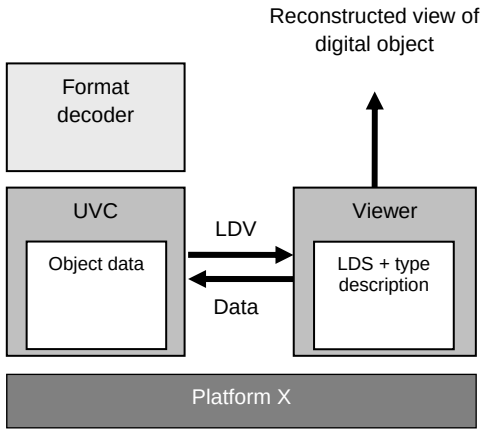

Figure 4.4: UVC-based preservation method

With this strategy future users should always be able to access and view the original object, although several steps must be taken at present and in the future. At preservation time, the official UVC specification must be preserved defining how the UVC should be build in the future. Also decoders must be developed; for each specific file format a decoder is needed. Furthermore, an LDS is required for each type of digital object, defining object types by image, sound, spreadsheet, text, etc. Of course the original objects should be preserved as well.

At retrieval time, future programmers have to redevelop the UVC, based on the UVC specification. Also, one or more viewers have to be created that translate the LDV into a representation of the original object.

The actions required at preservation and retrieval time reveal both the strengths and weaknesses of this strategy. In view of the extremely wide variety of file formats and types of objects, it will be necessary to develop a large number of format decoders and LDS descriptions if the UVC strategy is to be implemented as a means of providing durable preservation of digital objects. The success of the UVC strategy will depend on the extent to which it is accepted by the software and computer industry and the availability of format decoders and LDS descriptions. However, once a single LDS and decoder are developed the advantage becomes clear. Instead of performing migration steps for each individual object, the UVC can serve the need of all objects of the same format. It is also necessary to write a new emulator for each generation of hardware, but this should be relatively easy because the UVC is designed to be as simple as possible.

In 2001 the Digital Preservation Testbed of the Nationaal Archief developed and tested an initial version of the UVC to verify this approach against other strategies as migration and XML conversion [Testbed (2)]. With electronic spreadsheet documents as subject of the test, the UVC approach was stated less attractive for spreadsheets than migration and XML, although the UVC was not fully developed to show its potential. During 2003 and 2004 the KB and IBM continued the work on this approach and successfully developed an operational UVC for image formats. This approach works well for image formats [Wijngaarden] and can be applied to all other digital objects which do not contain behavioural aspects, but it is not suitable for digital objects that depend on their environment. To use the UVC for emulation, some major issues will have to be resolved. Apart from performance issues, an emulator should be written in UVC language and the UVC will have to be adjusted to be able to address more I/O functionality.

4.3. Conceptual model: modular emulation

Besides the evaluation of current emulators and approaches to apply emulation in a preservation context, a new approach for the realization of full emulation (emulation of hardware) has been developed. This approach, which should retain access to all kinds of digital objects over the long term, is a conceptual model and was initially intended to serve as opinion making and validation against other approaches. Later on, this model has evolved into a commonly defined scenario that forms the centre idea for development of the RWS emulator.

The principle of this model is based on the ideas of Rothenberg (RAND Corporation, USA) and Lorie (IBM Almaden Research Center, USA). In this, the use of a Virtual Machine (VM) and emulation specification comes from Jeff Rothenberg, whereas the universal platform and component library are abstracted from the UVC and its library of decoders by Raymond Lorie.

The model, as depicted in figure 4.5, shows a modular approach of emulation and is called *modular emulation model*. Modular emulation can be defined as:

> "full emulation of hardware by emulating the components of a hardware architecture as individual emulators and interconnecting them to form a full emulation process. In this, each emulator forms a distinct module of the total emulation process offering the possibility to make flexible configurations of different modules."

This modular approach comes close to the natural architecture of hardware (known as the Von Neumann architecture) and offers great flexibility in configuring new emulators by reusing modules as its individual hardware components.

The conceptual model consists of the following parts:

- Universal Virtual Machine
- Modular emulator
- Component Library
- Emulator specification document
- Controller
- Preservation Manager (optional)

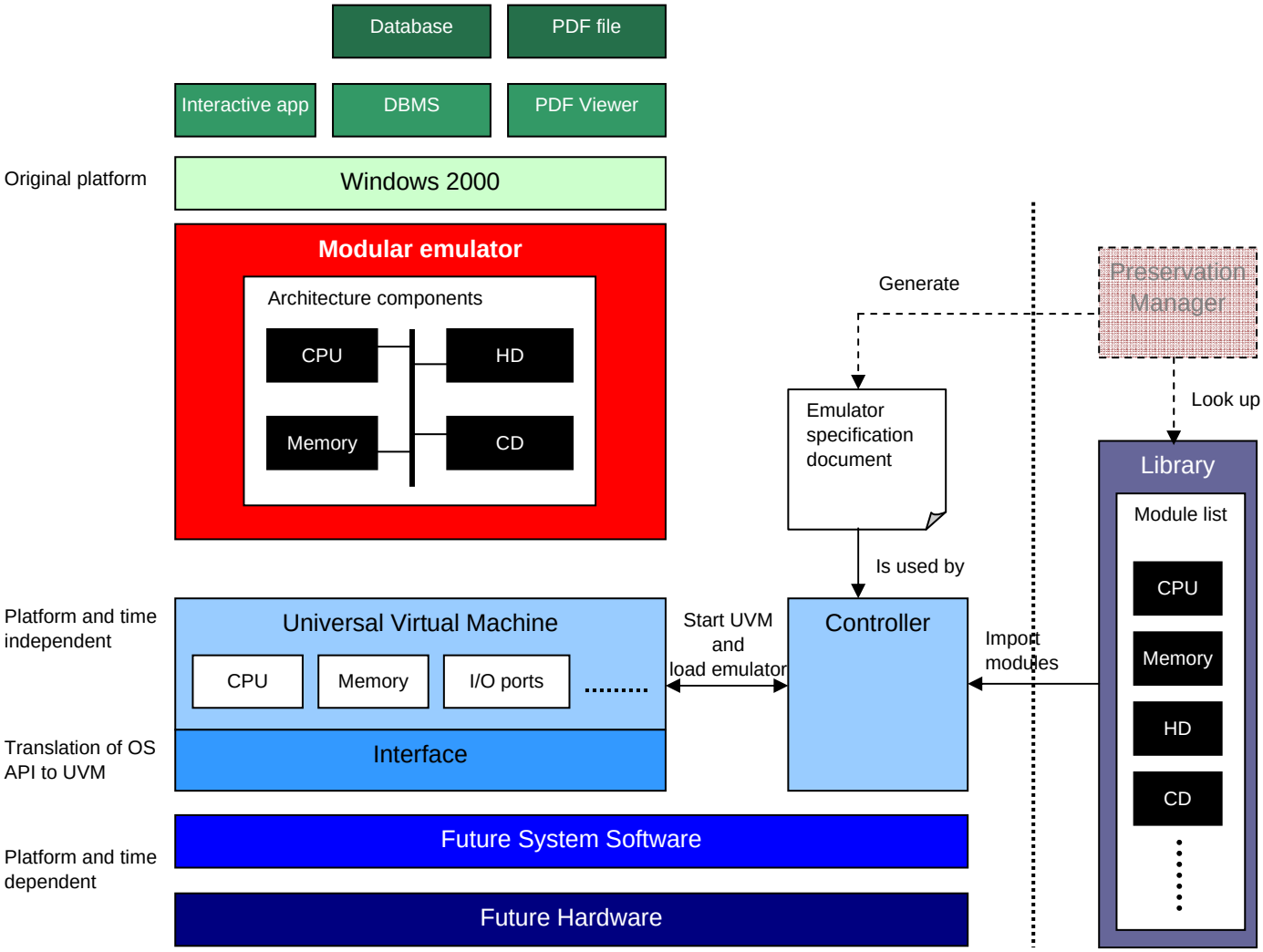

Figure 4.5: Conceptual model of modular emulation

4.3.1. Universal Virtual Machine

The basis of the modular emulation model is the Universal Virtual Machine (UVM). The UVM is a platform- and time-independent layer on top of the underlying future host platform. It could be seen as an advanced version of the UVC that not only supports a general purpose processor and memory, but also offers additional functionality for input and output (I/O) with peripheral devices between host and target platform. Therefore a universal I/O mechanism should be developed supporting I/O such as storage devices, screen output, networking and print facilities of the host platform towards the target platform.

4.3.2. Modular emulator

To execute the original software (OS and applications) on the UVM, an emulator is needed. The task of the emulator is to recreate the original hardware platform in software in such a way that the target software can run on it as it normally did on the original platform.

The emulator should be designed in a modular structure, in other words a modular emulator. The components of the original hardware platform are represented as modules in this structure. Each module emulates a specific hardware component in software on the UVM. For example, the functional behaviour of a graphics card is emulated by one single module. A recreation of the target platform can then be realized by combining all necessary modules following a hardware architecture. Normally this will be the Von Neumann architecture, but other (future) architectures should be possible. Interconnecting and controlling these modules should be done by the modular emulator.

4.3.3. Component Library

Building a modular emulator requires different modules. Therefore a library is needed, which contains all modules that can be used by the modular emulator. The library is the owner and manager of all modules. It should keep track of the available modules and organize it by type of components. These could be: CPU, memory, storage devices, network devices, etc. In a second phase, the type of components could be further refined by creating different implementations, e.g. for component CPU there is Intel Pentium, AMD Athlon, IBM PowerPC or Sun SPARC.

The level of detail at which a module is implemented is left up to the programmer. However, each module should recreate the functional behaviour of the actual hardware component as best as possible. All modules should be written in UVM language to make it run on the UVM platform. The library should offer a minimum set of modules to create at least one target platform, but can be expanded with more modules over time.

4.3.4. Emulator specification

Having a library of modules, a UVM and a modular emulator recreating the target platform, the emulator should be told which hardware architecture and modules are needed. Therefore, a so called emulator specification document should be written, defining a particular hardware architecture and its necessary components. Based on this document, the modular emulator can decide how the modules should be arranged and linked with the underlying UVM architecture.
Figure 4.6 defines a human readable emulator specification document. In this, it starts by telling that it uses the Von Neumann architecture, followed by a definition of each component. To offer a better structure of this document, it could also be written in XML, as shown in figure 4.7.

Computer architecture : Von Neumann

CPU

- Chipset : x86 compatible
- Clock speed : 700 MHz

RAM

- Capacity : 128 MB
- Speed : 133 MHz

Storage devices

Number of magnetic drives : 1
Number of optical drives : 2

Magnetic

- Type : hard disk
- Bus type : IDE
- Controller interface : ATA0 master
- Capacity : 20 GB
- RPM : 7200

Optical

- Type : DVD-ROM
- Bus type : IDE
- Controller interface : ATA0 slave
- Read speed : 10x
- Write speed : -

Optical

- Type : CD-ROM R/RW
- Bus type : IDE
- Controller interface : ATA1 master
- Read speed : 24x
- Write speed : 8x

Graphics adapter

- Bus type : AGP
- Capacity : 32 MB
- Max. resolution : 1024 x 768
- Color depth : 32 bit
- VESA support : yes
- 3D rendering support : no

Networking device

- Bus type : PCI
- Speed : 10/100 Mbit

Figure 4.6: human readable emulator specification document

In XML, this specification could look like:

```
<EMULATOR>
      <ARCHITECTURE>
            <NAME>Von Neumann</NAME>
            <TYPE>x86</TYPE>
            <COMPONENT>
                  <TYPE>CPU</TYPE>
                  <CHIP>x86</CHIP>
                  <SPEED>700</SPEED>
            </COMPONENT>
            <COMPONENT>
                  <TYPE>RAM</TYPE>
                  <CAPACITY>128</CAPACITY>
                  <SPEED>133</SPEED>
            </COMPONENT>
            …
      </ARCHITECTURE>
</EMULATOR>
```

Figure 4.7: emulator specification document in XML

4.3.5. Controller

A final essential part is the controller. The controller should start the UVM on the host platform and build a bridge between the I/O handling of the host platform and UVM. How this should be carried out is left up to the future programmer, because it depends on specific host platform characteristics. A second task of the controller is the delivery of the emulation specification document and modules at the modular emulator.

4.3.6. Preservation Manager (optional)

In theory, the emulation specification document could be automatically generated by the Preservation Manager (PM). When a user of the PM selects a viewpath for a particular digital information object, this path could be translated into an emulator specification whereby the components of the view path form the modules of the emulator. However, this is only possible if the view path is highly detailed (not only naming the hardware platform, but also describing the different internal components of it) and the PM is linked with the module library.

4.3.7. Practical steps

Several steps are required to take the presented conceptual model into practice. Best approach is to make a distinction between actions to be performed at preservation time and at retrieval time.

At preservation time

Step 1 – Define the UVM platform in an unambiguous specification document. I/O communication deserves extra attention, although it can not specified in detail because of its dependencies of unknown future hardware.

Step 2 – Create a module library, consisting of all necessary modules that represent at least one original platform. These modules should be specified and written using the UVM architecture. Tests should be done to evaluate the functional behaviour of each module.

Step 3 – Develop a modular emulator that can run correctly on the UVM, using the modules created in the component library.

Step 4 – Preserve the original digital objects to be run under emulation, together with the module library, emulator specification document and UVM specification document.

Step 5 – (Optional) Adjust the Preservation Manager to cooperate with the module library and automate the generation of emulator specification documents.

At retrieval time

Step 1 – Develop a UVM conform the UVM specification document. Implement a mechanism for I/O communication between the UVM and platform at that time.

Step 2 – Develop a program that controls the UVM and offers a way to send and receive information to and from the UVM.

Step 3 – Use or create an emulator specification document that describes the target platform (dependent on the available preservation manager functionality). If the module library is further expanded with more and improved modules, it is desirable to evaluate the specification document based on the current available modules.

Step 4 – Retrieve all necessary digital objects (or image file) from the digital archive and start the emulation process. The controller should then fulfill the following tasks:

1. Recognize the facilities the UVM supports (CPU, memory, storage, network, etc.).
2. Interpret the emulation specification document.
3. Decide which modules can be supported, if not available try compatible.
4. Retrieve the chosen modules from the library.
5. Link them to build the modular emulator.
6. Load the stream of objects (i.e. the image file or electronic documents).
7. Start the modular emulator with the digital objects to be reconstructed.

4.4. Environment and object preservation

Defining an emulation-based preservation strategy not only includes defining the process, but also organising the way the original objects and their environment are stored. Although the first objective of the project is to build an emulator, it is important to consider how the environment should be preserved to guarantee that the emulator is able to use this environment in the future. The environment not only includes (a virtual representation of) the original hardware platform, but also the system software (including drivers and configuration), applications and plug-ins.

Storage of these parts should be done in a way that satisfies the following criteria:
- Possibility to create, modify and emulate the environment.
- Comply with a “de jure” or “de facto” standard.
- Useful for reconstructing the digital objects of the scope: interactive multi-media applications, database systems and PDF documents.

The required information (system software, applications and data) for the emulator is normally offered by disk images. A disk image is a snapshot of the content of a partition on a physical storage device, like a hard disk, floppy disk or optical disk. A disk image is thus a virtual drive, containing the same information as the logical partition on the physical drive has. When an emulator is started, different disk images can be attached to it. Common are a hard disk image containing the OS and applications or a CD-/DVD-ROM disk image imitating the actual CD or DVD drive with optical disk inserted.

The disk image is stored as a large file on a real drive. The volume size of the disk image can be manually defined. It could be as large as the actual volume size of the device or even larger when the internal data within the disk image is compressed. Many different disk image formats exist based on two different approaches: raw or compressed images. Raw images are very similar with the actual content and layout of the physical drive and are also the same size (40GB hard disk is creates a 40GB disk image). A compressed image offers the benefit of much smaller disk images, but depends on a specific compression technique and makes the format more complex. Moreover, the compressed disk images should be extracted before they can be used, affecting the performance of the emulation process.

The content of a disk image depends on how the environment of the physical drives is arranged and whether or not applications and digital objects are included in the image as installed items. These items could also be stored in separate disk images. Different choices can be made about the definition of the disk images. For the emulation project choices can be made on a case-by-case basis.

4.4.1. Interactive multi-media applications

The interactive multi-media applications are supplied on CD- or DVD-ROM by publishers. An installation procedure is sometimes required to transfer the application from their carrier onto the software system environment on the hard disk before they can be executed. Installation of these kinds of applications should be rather simple, because any computer user must be able to install and work with it. The main difference of this type of object compared with database systems and PDF documents, is that the data is integrated with the application.

At the KB, each interactive multi-media application is installed on the RWS running a clean install of the Microsoft Windows 2000 operating system (OS) and some additional software. When the application is installed and operating correctly, a snapshot is taken from the hard disk. This is done by using the program PowerQuest (now owned by Symantec and called Drive Image), which makes a

compressed disk image of the hard disk's content as is, i.e. the OS plus installed interactive multi-media application, drivers and configuration. This disk image is then ingested into the e-Depot.

The original application can be accessed by retrieving the disk image from the e-Depot (by dissemination) and loading it onto an empty hard disk of the RWS or compatible computer system. Then the OS and subsequently the application can be executed. This process is depicted in figure 4.8.

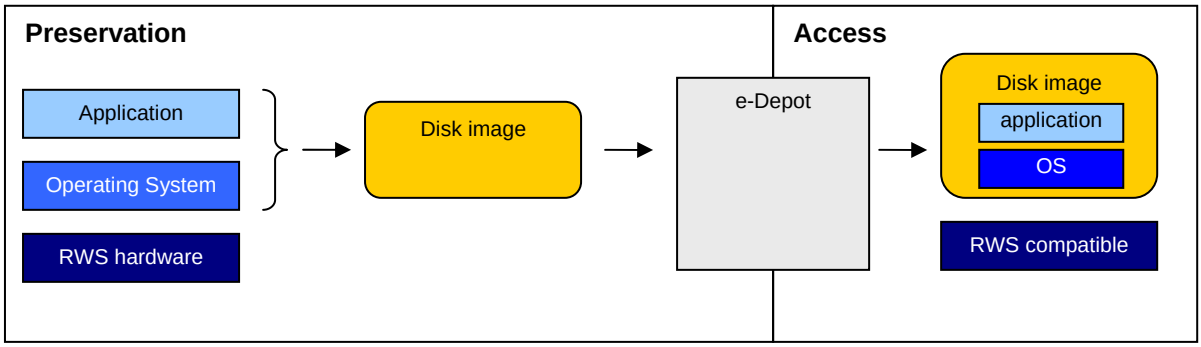

Figure 4.8: preservation of and access to interactive multi-media applications

Although this approach works well, the application and system software are integrated into one disk image that makes it impossible to separate the application from the system software afterwards. Separation could be useful when the OS becomes obsolete, even under emulation, and the application could be installed again on another OS. This offers the possibility of making on the fly configurations between OS and application. Figure 4.9 shows both choices for disk image storage.

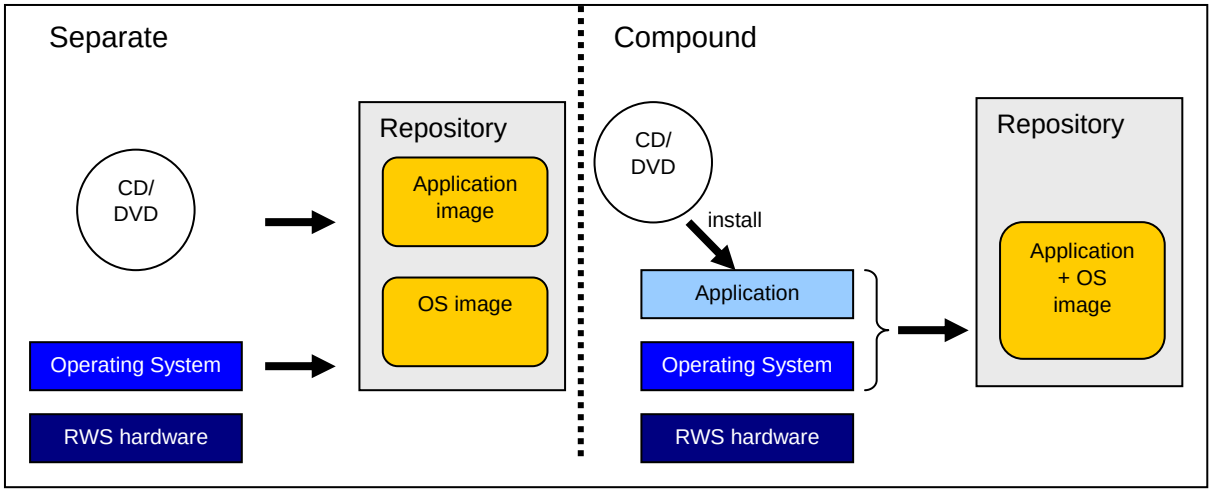

Figure 4.9: preservation of interactive multi-media applications

A second advantage is that the amount of storage space needed to store the objects decreases significantly. Storing a particular OS once offers better efficiency of the resources, especially when many different applications need to be preserved.

Additionally, if the original CD- or DVD-ROM is stored as a standardized disk image (e.g. ISO), it can be attached (mounted) to the emulated optical (virtual) drive letting the OS think that the original optical disk is inserted. This overcomes the problem of applications requiring the original CD- or DVD-ROM.

However, combining both OS and application together offers a more stable package when trying to access it. On forehand it can be guaranteed that this compound disk image worked without installation and configuration. This is not the case when separating the objects.

4.4.2. Database systems

Storage of database systems (DBS) requires a different approach than that of interactive multi-media applications. Although separation of OS and DBS offers the flexibility of combining different platforms with the same software, installation and configuration of a DBS requires substantial effort and knowledge. Moreover, the storage benefits are limited because of the limited number of database systems. Integration of OS and DBS seems therefore necessary.

As mentioned before, a database system mainly consists of three components:

1. Database : the actual content; the tuples or rows in the database tables.
2. DBMS : the database management system, computer program designed to manage a database and run operations on the data requested by users. It forms the heart of the database system.
3. Database application : computer software written to manage the data of a particular application or problem set. It normally consists of a user interface (forms) and scripts and uses the functionality of the DBMS to send and retrieve information to or from the database.

The arrangements of these components may vary for each database system. However, a rough classification can be made by dividing database systems in desktop (single user) database systems and complex (multi user) database systems (see figure 4.10).

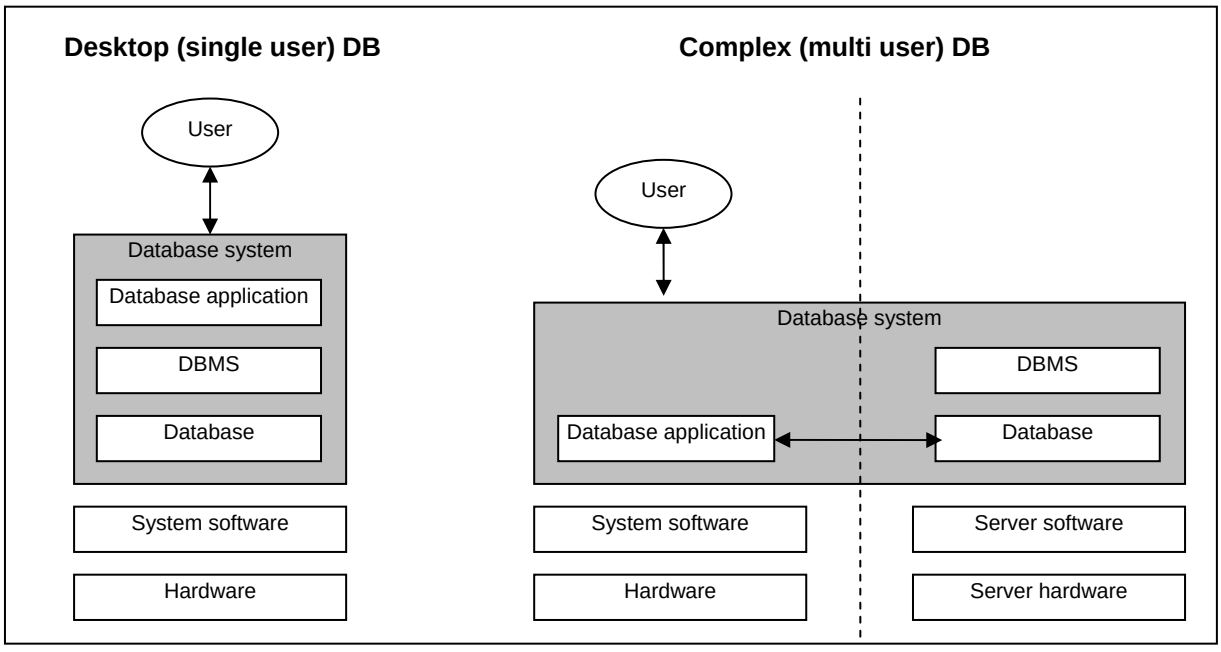

Figure 4.10: Database architecture for desktop (single user) and complex (multi user) database systems

Desktop (single user) database systems often exhibit the following characteristics:

- The DBMS can be used as a complete solution for both the database itself and the application.
- The database is often used by a single user at the time.
- Software of this nature is intended for end users, and consequently is easier to use than the software designed for the development of complex database systems. Although users need to receive training, this is insignificant in comparison with the amount of specialized training required for complex database systems.
- Users can migrate their database system to a different database system format or a new version of the same DBMS themselves.

Examples of desktop database systems are MS Access and dBase IV.

Complex (multi user) database systems often exhibit the following characteristics:

- The database and DBMS are separated from the application.
- The database is often used by multiple users at the time.
- Their design, construction, and maintenance require specialist skills.
- It is possible to convert the content of the database to another database format, or it can be migrated to a new version of the same DBMS. Specialized skills are also required for this conversion/migration.

Examples of complex database systems are Oracle products, Microsoft SQL Server and DB2.

Database systems created with a desktop database application like MS Access often consist of one single file (for MS Access an .mdb file) that contain not only the database content but also the scripts and forms. These files all use the same general purpose database program MS Access, which means that MS Access could be pre-installed on the OS while the database files can be stored separately. In this way, MS Access can be reused for every database file, as depicted in figure 4.11. For safety reasons, a disk image of the file system containing the installation files of the database program should also be preserved.

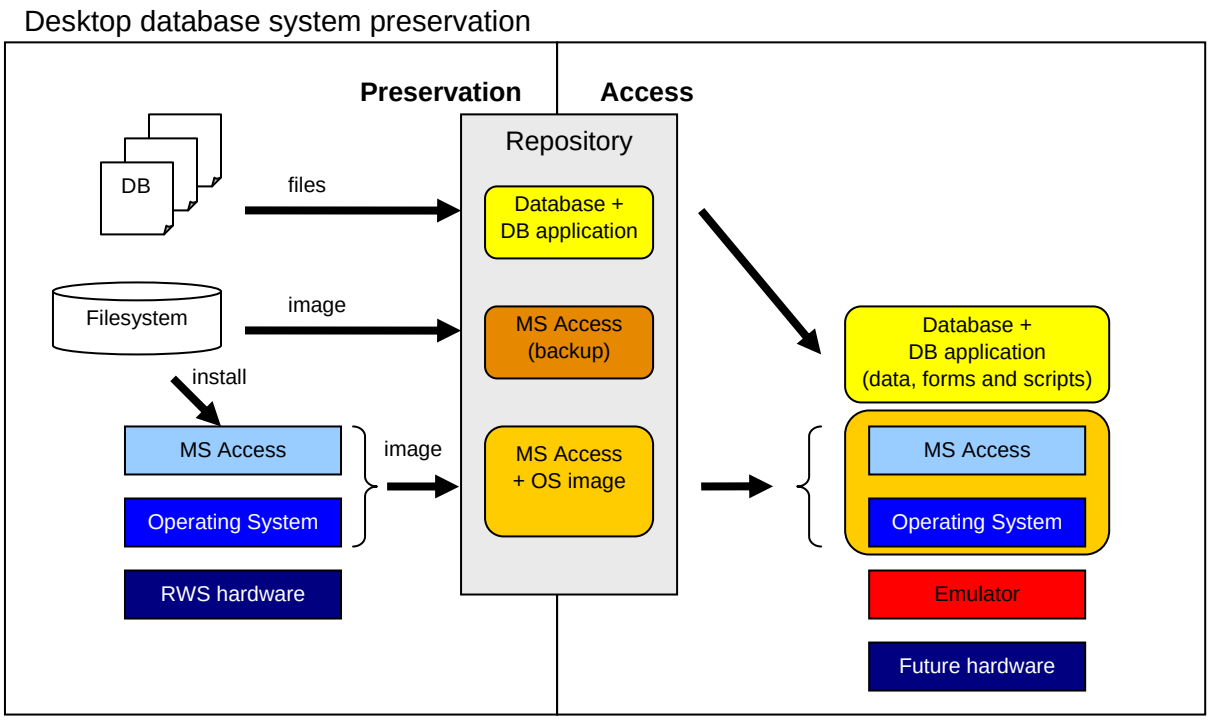

Figure 4.11: Example of preserving desktop database systems: MS Access

For complex database systems, separation of database application from DBMS and database enables the possibility to change, convert or replace the underlying DBMS and database while the database application remains the same. Although it offers flexibility to the system, each complex database is often custom-made. Applications designed to operate on one database cannot be used for a different database. Therefore, storage of complex databases made with Oracle should be done as a whole, thus: one disk image with OS, DBMS, database and database application (see figure 4.12). It must be noted that the client-server architecture should be realised on one single computer environment (running locally).

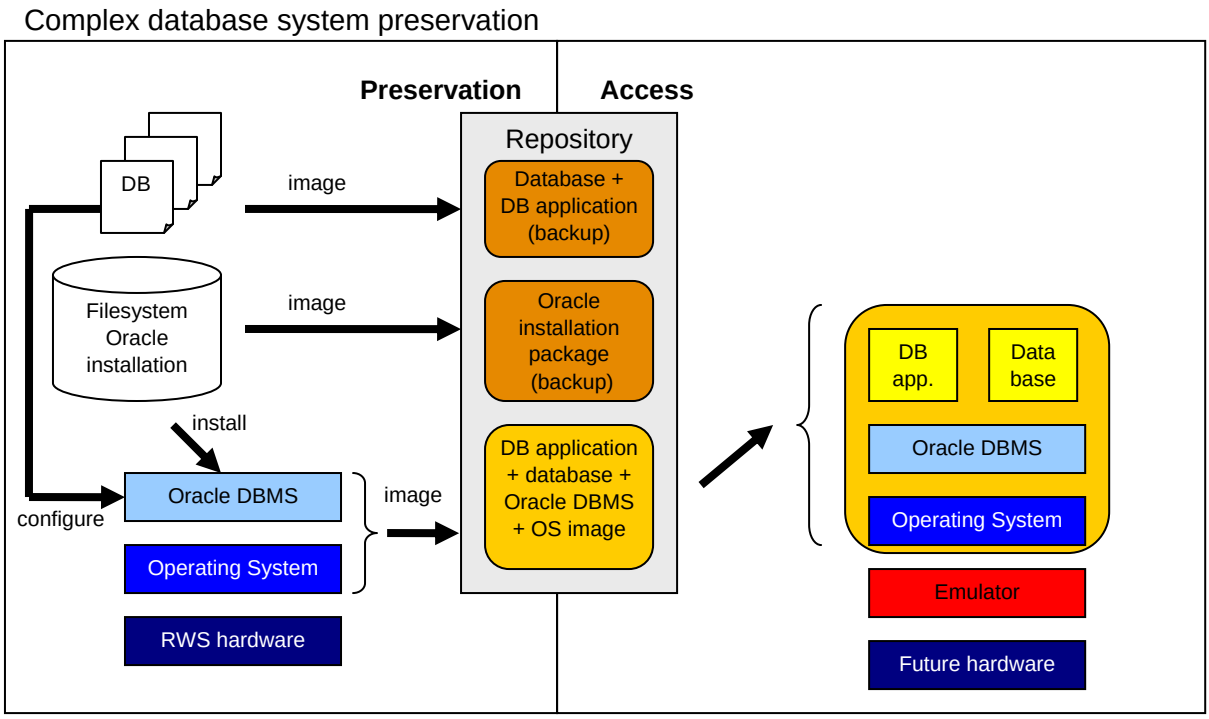

Figure 4.12: Example of preservation of complex database systems: Oracle Database

4.4.3. PDF documents

As with databases, PDF documents require an application before they can be viewed. This application, the PDF viewer, can be preinstalled on the OS. Separating the viewer from the OS does not offer great storage benefits and has the disadvantage that the viewer has to be installed every time a PDF document needs to be viewed. Figure 4.13 depicts the storage strategy for this kind of objects.

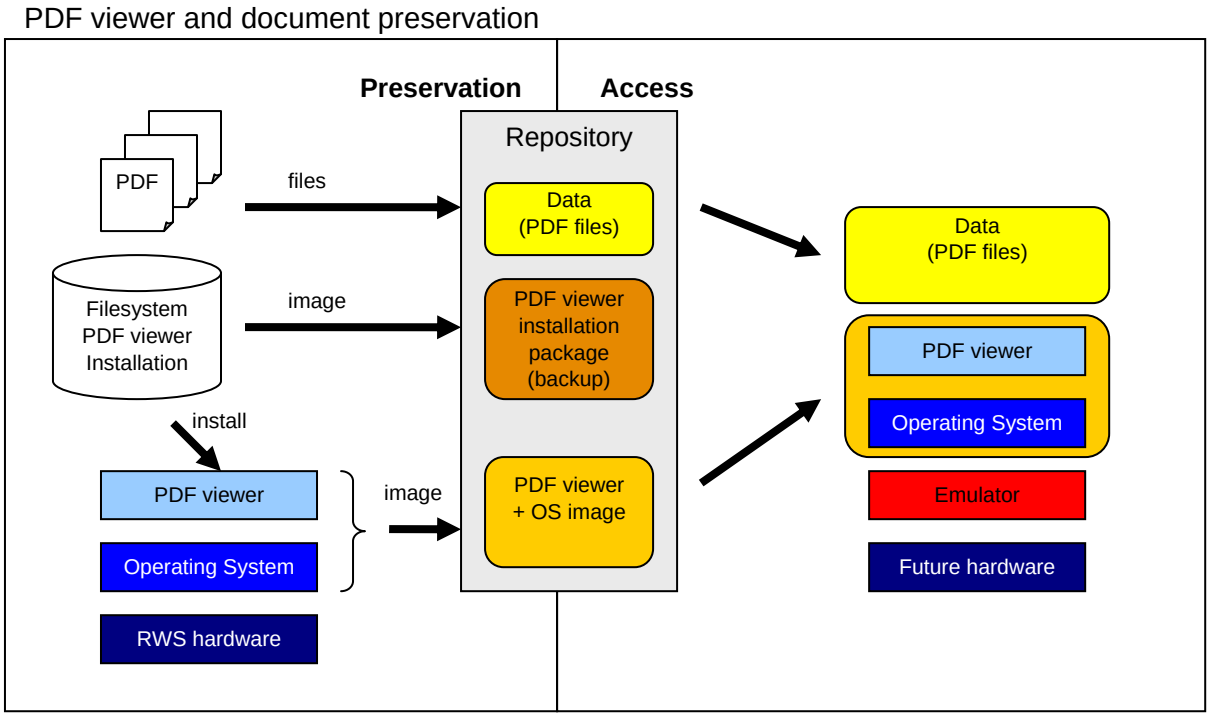

Figure 4.13: preservation of PDF documents and appropriate viewer

Table 4.1 summarizes the above considerations for storage and shows a possible disk image policy.

Digital object	Disk image policy
Interactive multi-media application	Two disk images: 1. Operating System 2. Installation package of interactive application
Database system	Desktop database system, two images: 1. Operating System + DBMS 2. Installation package of DBMS (backup) Database + DB application as separated files Complex database system, three images: 1. Operating System + DBMS + DB application + database 2. Installation package of DBMS (backup) 3. Database + DB application (backup)
PDF document	Two disk images: 1. Operating System + PDF viewer 2. Installation package of PDF viewer PDF documents as separated files
For all digital objects hold that besides the disk image(s) the original installation CD- or DVD-ROM should be preserved, together with a manual and a valid serial license key.	

Table 4.1: disk image policy

5. Planning and project organisation

The prior research led to good insight about the current state of emulation and appropriate emulators developed in the world. The next stage is to create a roadmap for the development of an emulator for digital preservation. In the following section, the actions in the KB/Nationaal Archief emulation project are described.

Five clusters of activities are defined:

5.1 Set up test environment
5.2 Define, describe and prepare test objects
5.3 Design and build a modular emulator
5.4 Reprogram the emulator to a Virtual Machine layer
5.5 Convert VM to a Universal Virtual Machine layer

5.1. Step 1: Set up test environment

The first step in the timeline for the emulation project is to set up a stable test environment: “an emulation lab”. This lab will be used for the testing of existing emulators as a follow-up of the research done at the KB, to describe and define test objects running on a Reference Workstation, to develop the emulator and to carry out experiments. The work in the testlab will have to prove the viability of the emulation strategy and will teach us about practical issues that will come up during development.

To set up this testlab, KB and Nationaal Archief will use an existing environment that has proved its use in practice and is build based on carefully chosen criteria: the Dutch Digital Preservation Testbed. This Testbed system has been developed for the Digital Preservation Testbed Project (2000-2003), a co-operation of the Nationaal Archief and the Dutch Ministry of the Interior and Kingdom Relations [Testbed (3)]. Experiments carried out in the Testbed system are documented according to a twelve-phase approach and are repeatable and can be checked by others. This system is very suitable as a testlab for the emulation project, but does require a technical update. The Testbed system is now five years old and in need of new hardware (while keeping the older workstations alive for test purposes and for consulting results of previous experiments) and system software. A standard Reference Work Station will be added to the Testbed system. To work with the Testbed system in the emulation project, the system will have to be connected to the Nationaal Archief network and a connection with the KB would be desirable. Since the Testbed is already webbased, this connection will allow KB- and Nationaal Archief- staff to access the system from their own workstations.

Activities to set up the Testbed system for the emulation project will include:

1. Add new computers to the existing ones; one to develop the emulator (which will have to be a high-speed, high-capacity pc), one as a server and one for carrying out the actual experiments.
2. Add a Reference Workstation to the Testbed environment.
3. Upgrade of the Testbed system with current software, including the current Oracle database version and other Oracle components.
4. Connect the Testbed system to the Nationaal Archief network.
5. Make a connection from the Testbed system to the KB-network.

5.2. Step 2: Define, describe and prepare test objects

The emulation project has a strong practical focus in that it will provide a working solution for the digital objects that are actually stored at the KB and Nationaal Archief. These objects have to be chosen carefully. After choosing a test set consisting of CD-ROM publications, pdf-journals and databases, these have to be tested and documented. This can be done by archive- and library-employees who are responsible for the storage of these collections.

A next step involves the storage and composition of the test objects. This includes the choice of disk image organisation. In chapter 4.4 an explanation of disk images and storage possibilities is given. One of the activities in the project will be to continue research on this topic, make a choice about the composition of the test objects and make a recommendation for storage.

The following questions should be answered:

1. What specific test object do we use for the project
2. What are the significant properties of these test objects when running on a RWS?
3. How do we document these significant properties?
4. Do we separate the environment software from the application?
5. What different file structures are there and which one do we choose for the project?
6. Which disk image format should be chosen?

5.3. Step 3: Emulate the RWS

To guarantee that the currently stored digital objects are still accessible in the future, we will recreate the original RWS environment. This will be done by using an emulator capable of offering the same aspects as the original RWS. As a result, the OS actually ‘thinks’ it is running on real RWS hardware.

Certain requirements of the RWS hardware platform have to be met. In short, it has to be an x86 platform with a certain amount of internal memory and hard disk space, a compatible display adapter, sound card and networking capabilities. Furthermore it should be able to attach and boot from a disk image. The specific RWS hard- and software configuration can be found in appendix A. For host-platform, we have to choose a platform that can run an emulator that meets these requirements.

In the previous chapters is explained how a modular design adds extra flexibility to the configuration of an emulator. In this way different emulators can be configured using and reusing emulator modules that find their analogy with real hardware components. The idea for the modular emulator has to be worked out and a technical design has to be written.

The preliminary research-study has shown that existing emulators are very promising, also for applying them for preservation. To build a modular emulator, the KB/Nationaal Archief will start with an existing emulator and look into possibilities to transform this into a modular emulator for preservation.

Activities to build the emulator can be separated into three groups: the architecture and technical design of the modular emulator, the choice of existing emulator, and programming the modular emulator.

1. Work out architecture for the modular emulator
2. Make a technical design for the modular emulator
3. Choose a host platform

4. Choose a programming language
5. Can we use an existing emulator?
6. Is this an open-source emulator and can we re-model this to a modular design?
7. What are the prospects of the emulator about current and future developments?
8. What modules can be re-used and what modules have to be developed?
9. How will the modules be managed?
10. Is there a relation with the Preservation Manager?

5.4. Step 4: Insert a Virtual Machine layer

The resulting emulator of the previous step is assumed to be compatible with the RWS, but is probably still bounded to a particular host platform (OS + hardware). To overcome platform-dependency a Virtual Machine (VM) has to be introduced. Several requirements can be defined for a VM:

- The VM has to support multiple host platforms.
- The VM has to be stable.
- It is preferred if the VM is a standard ("de facto" or "de jure").
- Compatibility over time is important.

The following questions have to be answered:

1. Does a VM exist that meets the requirements?
2. What is the trade-off between compatibility and authenticity?
3. How can we apply a VM to the current emulator?
4. Is the emulated platform still authentic with the original?
5. Does the performance of the emulator on the VM still meet the requirements?
6. If we choose the Java Virtual Machine as a host platform, the emulator has to be written in Java.
7. Would it be possible to port an emulator that is written in a certain language to Java?

5.5. Step 5: Convert VM to a Universal Virtual Machine layer

At this point the emulation process has two important aspects: it is flexible and platform-independent. The final step is to ensure that the emulated RWS platform keeps accessible over time, by adding a third aspect: time-independency. This can be achieved by transforming the Virtual Machine layer into a Universal Virtual Machine (UVM) layer.

Running a modular emulator on a Universal Virtual Machine, is the ultimate direction of the emulation project. However, designing and developing a UVM is a major effort, which cannot be part of this current project. Initiatives have started already (e.g. IBM's UVC) but have not yet reached a stage in which they can be used for implementation of full emulators. Therefore, this final step in building an emulator for digital preservation will be out of scope for this current project. If a UVM does exist, next issues become relevant:

Important characteristics of the UVM should be:

- The UVM should support multiple host platforms.
- The UVM should be stable.
- The UVM must be able to run the modular emulator.
- The UVM may expand (not change) over time, but has to maintain compatible.

The following questions have to be answered:

1. Is the UVM suitable to run the emulator?
2. If not, what should be changed?
3. How should the interaction between host and target platform be realised?
4. Is the emulated platform still authentic with the original?

5.6. Project organisation

The steps described above, are mostly chronological, with some interdependencies. After setting up the test environment, changes may still have to be made as a consequence of for instance the choice of test objects in step 2 or the choice for a programming language in step 3.

To carry out the project, a team should be set up that consists of a project leader, an architect, programmers, researchers and testers. The project leader, researchers and testers will be recruited from the KB and Nationaal Archief staff, architecture and programming can be tendered out.

A researcher is already available. As soon as a project leader is available, the rest of the team can be put together. To find an architect and programmers, a European tender procedure will be started. Setting up the test environment can start in April 2005, design and programming will take off in the summer. At the end of 2006, a working emulator, running on a virtual machine, should be delivered.

6. References

[Archiefwet] All Dutch government laws can be found on the website Overheid.nl. An extensive overview of the most relevant archival laws and legislation can be found on the website of the Rijksarchiefinspectie (central government archival inspectorate).

[Bochs project] Bochs, think inside the Bochs. http://bochs.sourceforge.net

[CAMiLEON (1)] CAMiLEON project, http://www.si.umich.edu/CAMILEON/

[CAMiLEON (2)] Holdsworth, D., Wheatley, P.R.: Emulation, Preservation and Abstraction. In: RLG Diginews, vol. 5 (4), (2001). http://www.rlg.org/preserv/diginews/diginews5-4.html

[CAMiLEON (3)] Holdsworth, D.: C-ing ahead for digital longevity. CAMiLEON project, University of Leeds, 2001. Available at: http://129.11.152.25/CAMiLEON/dh/cingahd.html

[Lorie] Lorie, R.A.: Long-term archiving of digital information. IBM Research report, IBM Almaden Research Center, San Jose, Almaden, 2000.

[Ockerbloom] Ockerbloom, J.M.: Archiving and preserving PDF files. In: RLG Diginews, vol. 5 (1), (2001). Available at: http://www.rlg.org/legacy/preserv/diginews/diginews5-1.html

[QEMU project] QEMU, CPU emulator. http://fabrice.bellard.free.fr/qemu/

[Rothenberg (1)] Rothenberg, J.: An Experiment in Using Emulation to Preserve Digital Publications. Koninklijke Bibliotheek, The Hague, The Netherlands, 2000. Available at: http://www.kb.nl/coop/nedlib/results/emulationpreservationreport.pdf

[Testbed 2003] Emulation: context and current status, Digital Preservation Testbed, The Hague, 2003. http://www.digitaleduurzaamheid.nl/bibliotheek/docs/white_paper_emulatie_EN.pdf

[Testbed (2)] From digital volatility to digital permanence± Preserving spreadsheets, Digital Preservation Testbed, The Hague, 2003. http://www.digitaleduurzaamheid.nl/bibliotheek/docs/volatility-permanence-spreadsh-en.pdf

[Testbed (3)] Digital Preservation Testbed project. http://www.digitaleduurzaamheid.nl/index.cfm?paginakeuze=298&categorie=6

[Virtual PC] Microsoft Virtual PC. Available at: http://www.microsoft.com/windows/virtualpc/default.mspx

[Virtual Server] Microsoft Virtual Server 2005. Available at: http://www.microsoft.com/windowsserversystem/virtualserver/default.mspx

[VMware] VMware. Available at: http://www.vmware.com

[Wikipedia (1)] Compatibility layer, in: Wikipedia.org, the free encyclopedia. http://en.wikipedia.org/wiki/Compatibility_layer

[Wikipedia (2)] List of emulators, in: Wikipedia.org, the free encyclopedia. http://en.wikipedia.org/wiki/List_of_emulators

[Wikipedia (3)] VMware, in: Wikipedia.org, the free encyclopedia. http://en.wikipedia.org/wiki/Vmware

[Wijngaarden] Van Wijngaarden, H., Oltmans, E.: Digital preservation and permanent access: the UVC for images. In: Proceedings of Imaging Science & Technology Archiving Conference, San Antonio, USA. Available at: http://www.kb.nl/hrd/dd/dd_links_en_publicaties/publicaties/uvc-ist.pdf

7. Glossary

Supported by Wikipedia.org. These terms are in the context of digital preservation.

API	:	the Application Programming Interface (API) is a set of definitions of the ways in which one piece of computer software communicates with another. It is a method of achieving abstraction, usually (but not necessarily) between lower-level and higher-level software.
Bytecode	:	an intermediate code, created after compilation of source code, but is more abstract than machine code. In Java, bytecode runs on a Java Virtual Machine offering better portability between different platforms without the need to recompile the source code.
Database	:	an information set with a regular structure. Today, databases represent mostly computerised data, but that is not a requirement.
Database application	:	computer software written to manage the data of a particular application or problem set. It uses the DBMS for manipulating the content of the database.
Database system	:	the set of all components that is necessary to operate (create, modify, view) on information to be stored in a database. These components are the database (containing the information), the DBMS and the database application.
DBMS	:	the Database Management System (DBMS) is a computer program (or more typically, a suite of them) designed to manage a database and run operations on the data requested by users.
Digital object	:	A digital object is made using a particular combination of hardware and software: the hardware platform, system and application software and one or multiple files. Each file consists of a series of ones and zeroes that is interpreted by a certain combination of hard- and software. The result of that interpretation is a unique representation that is called the digital object. Each digital object can be experienced by its structure, content, context, appearance and behaviour.
Host platform	:	the platform on which the emulator runs. In this document often defined as a future machine.
Disk image (installed)	:	A disk (partition) image is a file that contains a copy of all information stored on a disk. The image stores the installed operating system, all programs, and all documents and settings.

Level of fidelity	:	the accuracy with which the components (devices) of a computer are reproduced.
Modular emulation	:	full emulation of hardware by emulating the components of a hardware architecture as individual emulators and interconnecting them to form a full emulation process. In this, each emulator forms a distinct module of the total emulation process offering the possibility to make flexible configurations of different modules.
Multi-tier application	:	an application with an architecture that consists of different layers, each with its own responsibility. For example, a client/server architecture is two-tier.
Object type	:	a group of digital objects with the same logical structure.
Prototype	:	a test program, which is designed for demonstration and evaluation purposes. Multiple prototypes are developed during an incremental development process (prototyping) to refine the requirements, design and implementation of a product.
Reference Work Station	:	or RWS, is the predefined computer platform (hardware and software) used at the KB to capture and restore CD- / DVD-ROM publications.
Significant property	:	an important property of a certain digital object, as experienced by the user.
Target platform	:	the platform to be emulated.
Vernacular extraction	:	a migrated version of the (content of the) original digital object, without intermediate migrations.

Appendix A: Reference Workstation (RWS) PLATFORM-10

Software		
Application	*Version*	*Files*
MS Windows 2000 Professional	Service Pack 3	w2ksp3.exe
MS Direct X for NT	8	DX80NTeng.exe
Display Drivers (Compaq nVidia LP AGP GeForce2MX-400)	6.13.10.4103	41.03_win2kxp.exe
Intel Chipset		SP22545.exe
Intel Ide		SP21356.exe
Intel pro 1000T		pro2kxpm.exe
Keyboard		SP21795.exe (not used because it only enables the function keys on the top of keyboard)
Mouse		SP21424.exe
Sound		SP22286.exe
Additional Software		
Application	*Version*	*Files*
MS Internet Explorer	5.50.4134.0600	ie552000 directory, setup is started via ie5setup.exe
MS Windows Media Player	7.1	mp71.exe
MS SysPrep (in deploy.cab)	1.1	W2KCD:\support\tools\deploy.cab and Windows 2000 System Preparation Tool (version 1.1) from Microsoft (sysprep_update)
Power Quest Drive Image Pro	4.0	Drive Image Pro 4.0 directory, subdirectory dp40en, subdir setup, contains setup.exe
Adobe Acrobat Reader	5.01	Acrobat Reader 5.01 directory, rp501enu.exe (only used on development to read the documentation)
Java Plug-in for the browser [=IE]	1.3.1_02 Standard Edition International	j2re-1_3_1_02-win-i.exe
InterVideo WinDVD	2000	Install CD delivered with DVD-ROM Drive (COMPAQ JLMS DVD-ROM LTD-1665)
Paragon CD-ROM emulator	2.5 Personal Edition	Install cd : iso ‘paragon cd.iso’
Hardware		
Component	*Specification*	
Main board	Compaq EVO D510 CMT	
Processor	Intel Pentium 4 system running at 2,4GHz	
Internal memory	1 GB RAM	
Hard disk devices	2 internal disks: 40 GB Maxtor 6E040L0 40 GB WDC WD400BB-60CJA0	
Display adapter	nVidia GeForce4 MX 420 display adapter Fill Rate : 1 Billion Texels/Sec. Triangles per Second : 31 Million Memory Bandwidth : 2.7GB/Sec.	

	Maximum Memory : 32MB
Sound device	SoundMAX AC97 Integrated Digital Audio
Optical storage devices	DVD-ROM Drive (COMPAQ JLMS DVD-ROM LTD-1665), Region 2
Floppy disk drives	3.5”floppy drive
Ports (external)	4 Universal Serial Bus ports (USB) 2 serial communications ports (COM) 1 parallel communications port (LPT)
Network adapters	100 Megabit Ethernet network adapter 1 Gigabit Ethernet network adapter
Monitor	CTX S500B *Viewing* Display Technology : Active Matrix TFT Panel Display Panel : 15" (Diagonal) Resolutions : 640 x 350 @ 70Hz 720 x 400 @ 70Hz 640 x 480 @ 60Hz, 72Hz, 75Hz 800 x 600 @ 60Hz, 72Hz, 75Hz 1024 x 768 @ 60Hz, 70Hz, 75Hz 1024 x 768 (Max) Contrast Ratio : 400:1 Brightness (Typical) : 250 cd/m2 Viewing Angle (H/V) : 120/100 (degrees) Pixel pitch : 0.297 mm Scanning characteristics : 30kHz - 60kHz (Hor. Freq.), 58Hz ~ 75Hz (Vert. Freq.) Response time : tr:13ms / tf:27ms Maximum color support : 16.7 million *Bodywork* Cabinet Color(s) : Black VESA compliant : yes Plug & Play : yes Power usage : 35W maximum Dimensions : 14.65" W x 14.02" Hx 6.77" D Weigth : Net: 8.16 lbs. Gross: 14.1 lbs Inputs signals : Video= RGB Analog (0.7Vp-p) Sync= H/V Separated (TTL) User controls : Front Panel Controls: Power Switch, LCD Indicator, ESC, Up, Down, Enter On-screen menu controls : Contrast, Brightness, Auto Tune, Color Temp, Size, Phase, Focus, Dithering, Text/Gfx, Position, Languages, Recall Environments : PC and Mac *miscellaneous* Safety regulations : FCC Class B, UL, cUL, CE, TCO'99
Keyboard	Compaq US keyboard for Microsoft Windows
Mouse	Compaq PS/2 (Logitech) mouse, wheel, wired, mouse ball

More information on the specification of the RWS:
Graphics card -> http://www.nvidia.com/page/geforce4mx.html
Sound card -> http://www.digit-life.com/articles/inteld815efvac97/
Monitor -> http://www.ctxtec.com/products/lcd_s500b.htm

Appendix B: emulator platform specifications

QEMU

The QEMU emulator includes the following characteristics:

- X86 CPU (16 and 32 bit), PPC (32 bit), ARM 7, SPARC V8
- FPU and MMU
- i440FX host PCI bridge and PIIX3 PCI to ISA bridge
- RAM
- Cirrus CLGD 5446 PCI VGA card or dummy VGA card with Bochs VESA extensions (hardware level, including all non standard modes)
- PS/2 mouse and keyboard
- Floppy disk
- 2 PCI IDE interfaces
- Hard disk
- CD-ROM
- NE2000 PCI network adapters
- Soundblaster 16 bit
- Serial ports

Bochs

The Bochs emulator includes the following characteristics:

- 386, 486, Pentium, Pentium Pro (incomplete) and AMD x86-64 (incomplete) emulation.
- FPU
- RAM
- Enhanced BIOS
- Cirrus CLGD 5446 PCI VGA card or dummy VGA card with Bochs VESA extensions
- PS/2 mouse and keyboard
- Floppy disk
- Multiple ATA channels
- Hard disk
- CD-ROM
- NE2000 network adapter
- Soundblaster 16 bit
- Parallel port
- Serial port
- Gameport
- PCI (incomplete)
- USB (incomplete)